МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЧЕРКАСЬКИЙ ДЕРЖАВНИЙ БІЗНЕС-КОЛЕДЖ

кафедра комп’ютерної інженерії та інформаційних технологій

КВАЛІФІКАЦІЙНА РОБОТА

на тему
Побудова та балансування відеоігрових економік

Виконав: студент групи 1П-20
Спеціальності
121 – «Інженерія програмного забезпечення»

Олександр ПАНЧЕНКО

Керівник:
Станіслав МАРЧЕНКО

Черкаси 2024

ЧЕРКАСЬКИЙ ДЕРЖАВНИЙ БІЗНЕС-КОЛЕДЖ

Кафедра комп'ютерної інженерії та інформаційних технологій
(повна назва випускової кафедри)

Спеціальність 121 "Інженерія програмного забезпечення"
(шифр і назва спеціальності)

Освітня програма Інженерія програмного забезпечення
(назва освітньої програми)

ЗАТВЕРДЖУЮ

Завідувач кафедри
комп'ютерної інженерії та
інформаційних технологій
(назва кафедри)
_____ Хотунов В.І.
(підпис) (ПІБ)
«_____»_____2023 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Панченку Олександру Ігоровичу
(прізвище, ім'я, по батькові студента)

1. Тема кваліфікаційної роботи Побудова та балансування відеоігрових економік
Керівник роботи Марченко Станіслав Віталійович, спеціаліст І категорії
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом закладу вищої освіти від " 13 " жовтня 2023 року № 65У.
2. Строк подання студентом випускної роботи 03.06.2024
3. Вихідні дані до кваліфікаційної роботи мова програмування C#, ігровий рушій Unity, універсальна мова моделювання (UML), сервіс побудови інтерактивних діаграм Machinations.
4. Зміст кваліфікаційної роботи (перелік питань, які потрібно розробити) огляд предметної області (базові поняття в контексті ігрової економіки, особливості побудови ігрових економік залежно від жанру гри, сучасний стан розроблення питання балансування ігрових економік), проєктування ігрової економіки (аналіз вимог до програмного забезпечення, нотація Machinations для проєктування ігрової економіки, модель ігрової економіки в Machinations), впровадження моделі ігрової економіки (розгортання програмного забезпечення, побудова демонстраційного прототипу гри, ігрова економіка в дії).
5. Дата видачі завдання 15.09.2023р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Терміни виконання етапів	Примітка про виконання з підписами наукового керівника і студента
1	Вступ	20.10.2023	
2	Розділ 1. Огляд предметної області	22.12.2023	
3	Розділ 2. Проєктування ігрової економіки	15.03.2024	
4	Розділ 3. Впровадження моделі ігрової економіки	15.05.2024	
5	Висновки	17.05.2024	
6	Оформлення випускної роботи (чистовий варіант)	27.05.2024	
7	Здача випускної роботи на кафедру для рецензування (за 14 днів до захисту)	31.05.2024	
8	Перевірка випускної роботи на наявність ознак плагіату (за 10 днів до захисту)	03.06.2024	
9	Подання випускної роботи на затвердження завідувачу кафедри (за 7 днів до захисту)	06.06.2024	

Студент _____ Панченко О.І.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Марченко С.В.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Кваліфікаційна робота присвячена розробці та балансуванню економічної моделі для відеоігор, зокрема ігор жанру Tower Defense. Метою роботи було створення збалансованої ігрової економіки, яка забезпечує справедливий та захоплюючий геймплей, стимулює інтерес гравців і підтримує тривалу взаємодію з грою.

У роботі розглянуто основні елементи ігрової економіки, включаючи ресурси, балансування, механізми отримання та витрати ресурсів. Проведено аналіз сучасних методів балансування ігрових економік, включаючи використання математичних моделей, аналіз даних гравців та штучний інтелект. Сформульовано завдання на розробку економічної моделі з врахуванням вимог до системи та очікуваних результатів.

Було розроблено концептуальну схему економічної моделі, яка включає основні компоненти та їх взаємодію. Описано інструментальні засоби для проектування, розробки та реалізації економічної системи гри. Впроваджено модель ігрової економіки, яка включає механізми генерування і витрати ресурсів, необхідних для будівництва та підтримки захисних споруд.

Робота зробила внесок у розвиток ефективних методик розробки ігрових економік, що може бути корисним для розробників ігрових продуктів та IT-команд у покращенні їхніх процесів проектування та впровадження ігрових економічних систем.

ABSTRACT

The qualification work is devoted to the development and balancing of an economic model for video games, in particular Tower Defense games. The aim of the work was to create a balanced game economy that provides fair and exciting gameplay, stimulates player interest and supports long-term interaction with the game.

The paper considers the main elements of the game economy, including resources, balancing, mechanisms for obtaining and spending resources. An analysis of modern methods of balancing game economies, including the use of mathematical models, player data analysis, and artificial intelligence, is carried out. The task of developing an economic model is formulated, taking into account the requirements for the system and the expected results.

A conceptual scheme of the economic model was developed, which includes the main components and their interaction. The tools for designing, developing and implementing the economic system of the game are described. A game economy model has been implemented, which includes mechanisms for generating and consuming resources necessary for the construction and maintenance of defense structures.

The work has contributed to the development of effective game economy development methodologies, which can be useful for game developers and IT teams in improving their processes of designing and implementing game economic systems.

ЗМІСТ

ВСТУП

Актуальність обраної теми. У сучасному світі індустрія відеоігор швидко розвивається, ставлячи перед розробниками складні завдання щодо створення захопливого та збалансованого геймплею. Однією з ключових складових успіху будь-якої гри є її економічна модель, яка забезпечує цікавий та справедливий ігровий процес. Економіка гри визначає, як гравці отримують, витрачають і обмінюють ресурси, що впливає на їхній прогрес і загальне задоволення від гри.

Забезпечення стійкого розвитку галузі відеоігор вимагає ефективного управління економічними аспектами гри. Правильне балансування економічних систем допомагає зберегти інтерес гравців, залучити нових користувачів та забезпечити довгостроковий успіх гри.

У відеоіграх часто спостерігається дисбаланс у розподілі внутрішніх ресурсів, таких як валюта, предмети або можливості, що може призвести до переваги деяких гравців над іншими та порушити баланс. Це, в свою чергу, може спричинити незадоволення серед користувачів. У цьому контексті побудова та балансування відеоігрових економік стає критично важливою для забезпечення стійкого розвитку галузі.

Мета дослідження. Мета проєкту полягає в побудові інтерактивної моделі збалансованої економіки для гри в жанрі Tower Defence та реалізації демонстраційного прототипу засобами ігрового рушія Unity. Особлива увага приділятиметься методам балансування ігрових економік, які забезпечують оптимальний розподіл ресурсів і створюють виклики для гравців. У роботі також розглядаються сучасні підходи та інструменти, зокрема інструмент Machinations, що використовується для моделювання ігрових економік.

Об’єкт дослідження. Об’єктом дослідження є ігрові економіки та підходи до їх балансування. Це включає аналіз систем виробництва, розподіл ресурсів, механізми торгівлі, валютні системи, ціни та будь-які інші елементи, які впливають на економічну динаміку у віртуальному світі гри.

Предмет дослідження. Предметом дослідження виступають методи й технології моделювання і впровадження збалансованих економічних систем у відеоіграх в жанрі Tower Defence. У цьому контексті моделюються механіки виробництва, розподілу та використання внутрішніх ресурсів, цінова політика, систему винагород та інші елементи, що впливають на економічний аспект гри.

У межах кваліфікаційної роботи обрано до розгляду та реалізації такі завдання:

1) дослідження та порівняння різних економічних моделей, що використовуються у відеоіграх;

2) вироблення підходів для забезпечення балансу ресурсів та стабільності для гри в жанрі Tower Defence;

3) створення інтерактивної моделі для оцінки ефективності використання розробленої економічної системи гри та визначення її впливу на геймплей;

4) розробка та впровадження економічної системи гри з урахуванням потреб та особливостей конкретного проекту.

РОЗДІЛ 1
ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Базові поняття в контексті ігрової економіки

Економіка у відеоіграх має життєво важливе значення для формування досвіду гравця та впливу на його рішення в ігровому процесі. Інтеграція валюти, ресурсів і торгових систем може суттєво вплинути на мотивацію гравця, його прогрес і загальне задоволення від гри. Від управління ресурсами до купівлі та продажу предметів – дизайн внутрішньоігрової економіки може суттєво впливати на взаємодію гравців, проходження гри та загальне враження про неї.

Розуміючи ці базові концепції, розробники ігор можуть створити динамічну та цікаву економіку, яка посилює занурення гравця в гру та задоволення від неї [2].

Ігрова економіка – це віртуальна економіка, яка налаштовує всі ігрові цикли в грі (валюти, часові цикли, здоров’я гравця, рівні, ціноутворення тощо). Різні ігрові економіки структурують поведінку інших гравців у межах однієї гри. Створення хорошої гри передбачає співпрацю кількох команд: художників, які створюють зовнішній вигляд, геймдизайнерів, які придумують механіки та сюжет, і програмістів та QA, які забезпечують код без помилок. Балансування ігрової економіки є ключовим для того, щоб все це працювало, і, в ідеалі, цим повинен займатися дизайнер ігрової економіки, який збалансовує всі числові показники в грі, пов’язує їх один з одним, щоб побудувати надійну ігрову економіку [4].

Внутрішньоігрова економіка працює подібно до економіки реального світу, де гравці заробляють і витрачають віртуальну валюту (наприклад, золото, монети або бали) в грі. Валюти є основою будь-якої реальної сучасної економіки, оскільки обміни здійснюються через них [5]. Гравці можуть заробляти валюту, виконуючи завдання та місії або продаючи предмети іншим гравцям. Потім вони мають змогу витратити цю валюту на ігрові предмети, апгрейди або послуги.

Внутрішньоігрова економіка може також включати ринок, де гравці купуватимуть і продаватимуть предмети, встановлюючи власні ціни на основі

попиту і пропозиції. Це створює економіку, керовану гравцями, де ціни можуть коливатися залежно від ринкових умов.

Гравці також можуть брати участь у віртуальній роботі або бізнесі в грі, щоб заробляти валюту, подібно до реальної зайнятості. Це може включати такі види діяльності, як фермерство, видобуток корисних копалин, ремесло або управління магазином.

Прикладом гри з реалізованою внутрішньоігровою валютою є World of Warcraft, в якій гравці можуть заробляти золото, виконуючи квести та дії в грі, досліджуючи величезний віртуальний світ, і витрачати зароблене на купівлю та покращення спорядження для свого персонажа. Водночас, у грі FarmVille гравці можуть заробляти монети, збираючи врожай та виконуючи завдання на своїй віртуальній фермі.

Загалом, внутрішньоігрова економіка надає гравцям можливість взаємодіяти, торгувати та конкурувати один з одним, додаючи глибини та складності ігровому досвіду. Ознаки гарно спроєктованої ігрової економіки включають здатність стимулювати гравців продовжувати гру, заробляти нагороди й рухатися далі [8]. Ефективна економіка гарантує, що всі гравці, незалежно від рівня майстерності або стилю гри, мають можливість досягти успіху і прогресувати в грі. Хороший дизайн економіки дозволяє гравцям робити осмислений вибір щодо того, як витрачати свої ресурси та просуватися в грі.

Економіку також можна використовувати для сприяння соціальній взаємодії між гравцями, наприклад, через торгівлю, спільне використання ресурсів або участь у спільному ігровому процесі. У багатьох безкоштовних іграх економіка покликана заохочувати гравців витрачати реальні гроші на внутрішньоігрові покупки, такі як аксесуари, бонуси або інші віртуальні предмети. Добре продумана економіка може допомогти продовжити життя гри, надаючи гравцям постійні виклики та цілі, до яких вони мають прагнути [1].

Балансування відеоігрових економік є критично важливим процесом у розробці гри, що забезпечує справедливий і захоплюючий ігровий досвід для всіх гравців. Важливість балансування полягає в досягненні рівноваги між

винагородами та витратами, що дозволяє гравцям відчувати прогрес і задоволення від гри.

Під час балансування необхідно враховувати різні аспекти, такі як доступність ресурсів, складність отримання цих ресурсів, а також способи їх витрат. Наприклад, якщо ресурси надто легко отримати, гравці можуть швидко втратити інтерес до гри. Навпаки, якщо ресурси надто складно здобути, це може призвести до розчарування і відтоку гравців.

Крім того, балансування включає аналіз динаміки попиту і пропозиції в грі, запобігання інфляції та дефляції, а також регулювання внутрішньоігрових цін. Важливо також враховувати різні стилі гри та рівні майстерності гравців, щоб забезпечити рівні можливості для успіху.

Для досягнення оптимального балансу розробники використовують моделювання та тестування економічних систем, коригуючи параметри на основі зворотного відгуку від гравців та аналітичних даних. Успішно збалансована економіка сприяє тривалому інтересу до гри, підтримуючи мотивацію гравців досягати нових цілей і продовжувати взаємодію з ігровим світом.

Хоча в невеликих компаніях створення ігрової економіки часто стає обов’язком геймдизайнера, економічна експертиза необхідна для балансування: вона дозволяє бачити загальну картину і проводити паралелі з реальним життям, використовуючи різні наукові методології [4].

Ключові компоненти ігрової економіки включають ресурси, балансування, поглиначі та джерела, інфляцію та дефляцію, прогрес гравця, зворотний зв’язок і адаптивність. Ресурси охоплюють ігрову валюту, предмети, очки досвіду та інші елементи, які гравці можуть заробляти, обмінювати або витрачати в грі. Балансування економіки передбачає забезпечення справедливого розподілу ресурсів і надання гравцям можливості заробляти або купувати ресурси різними способами. Поглиначі представляють собою способи витрати ресурсів гравцями, наприклад, на купівлю предметів або покращення спорядження, тоді як джерела визначають методи заробітку ресурсів, такі як виконання квестів або перемога над ворогами .

Інфляція та дефляція є важливими аспектами, які дизайнери повинні враховувати, щоб розуміти, як економіка розвиватиметься з часом, як можуть змінюватися ціни, як ресурси можуть ставати більш чи менш дефіцитними і як новий контент може впливати на економіку. Прогрес гравця має бути підтриманий економікою, яка забезпечує чіткі цілі та винагороди, стимулюючи гравців продовжувати грати та інвестувати в гру. Зворотний зв'язок в ігровій економіці дозволяє гравцям бачити вплив їхніх рішень і дій на ігровий світ, допомагаючи їм зрозуміти причинно-наслідкові зв'язки в економіці.

Адаптивність є ключовою рисою хорошого економічного дизайну, яка передбачає гнучкість і здатність адаптуватися до змін у поведінці гравців, ігровому середовищі або оновлень контенту. Це забезпечує стабільність та динамічність ігрової економіки, підтримуючи інтерес гравців і їхнє залучення до ігрового процесу [2].

Економічна діяльність являє собою процес використання економічних ресурсів і охоплює різні види виробництва, розподілу, обміну та споживання, що здійснюються для задоволення попиту. Розглянемо п'ять категорій абстрактної економічної діяльності ретельно розроблені для того, щоб охопити весь необхідний і розширюваний зміст більшості сучасних економічних систем MMO.

Під завданням (Task) в широкому сенсі розуміється будь-яка виробнича поведінка, в якій гравці безпосередньо отримують ресурси завдяки своїй праці в грі, наприклад, збирання ресурсів, проходження підземель та інший ігровий процес у MMO.

Модернізація (Upgrade) в широкому сенсі означає будь-яку споживчу поведінку гравців, спрямовану на покращення своїх можливостей шляхом споживання відповідних економічних ресурсів у грі, наприклад, підвищення рівня персонажа, спорядження чи зброї.

Під аукціоном (Auction) в широкому сенсі розуміється будь-яка торговельна поведінка, що передбачає вільну торгівлю між гравцями у грі. Всі

економічні ресурси, якими можна торгувати можуть бути предметом торгівлі, структурованої як безперервний подвійний аукціон.

Покупки (Shop) в широкому розумінні – це будь-яка торговельна поведінка, коли гравці безпосередньо купують товари в ігрових центрах або у неігрових персонажів (NPC) у грі.

Поповнення рахунку (Recharge) в широкому сенсі відноситься до будь-якої поведінки на валютному ринку, коли гравці здійснюють платежі в грі, наприклад, покупки в додатку (In-App Purchases, IAP) [11].

Інвестиційні ресурси в економічній структурі гри впливають на швидкість, з якою гравець отримує основну цінність гри. Такі ресурси повинні бути виражені в термінах часу, а потім обмежені для підтримки балансу. Важливо не мати інвестиційних ресурсів, які можуть приносити довготривалі дивіденди під впливом випадкових чинників [4].

Існують специфічні ресурси, які не можуть суттєво впливати на розвиток персонажа – наприклад, одноразові бустери, які допомагають гравцям завершити рівень трохи швидше. Хоча ефект від таких бустерів здебільшого незначний, їх варто брати до уваги та оцінювати з точки зору часу.

Неінвестиційні ресурси не впливають на розвиток гравця. Наприклад, скіни, які просто візуально радують гравців. Однак їх варто враховувати, оскільки гравці витрачають на них гроші, подібно до предметів розкоші, які люди купують у реальному житті. Іноді навіть досвід персонажа може бути ресурсом. Його не можна витратити, за нього не можна нічого купити, але він може бути життєво важливим для прогресу гри. Наприклад, поки гравець не отримає певну кількість досвіду, він не зможе отримати певний апгрейд. Таким чином, ці очки досвіду стають ресурсом, який може бути в дефіциті [4].

Зазвичай геймдизайнер під час створення геймдизайн документу (GDD) описує ресурси всіх типів. Однак, дизайнер ігрової економіки все одно повинен ретельно перевірити всі механіки, щоб виявити будь-які додаткові, не враховані ресурси. Ресурси в грі можна отримати з різних джерел: перемоги, ігрові досягнення, бонуси або навіть періодичні винагороди. Спосіб, у який гравці

накопичують ці ресурси, може дуже різнитися. Дехто може покладатися на послідовний ігровий процес, повільно накопичуючи ресурси з часом, тоді як інші можуть застосовувати стратегії великих перемог, здобуваючи велику кількість ресурсів за короткий період.

Розробники ігор повинні враховувати ці методи здобуття ресурсів і те, як вони узгоджуються із загальним ігровим досвідом та ігровою механікою. Наприклад, чи заохочує гра ризиковані, високоприбуткові стратегії, чи вона винагороджує стабільний прогрес?

Гравці можуть витрачати свої ресурси різними способами: робити ставки, купувати предмети в ігровому магазині або інвестувати в розвиток гри. Швидкість і спосіб витрачання коштів можуть багато чого розповісти про толерантність до ризику та ігровий стиль гравця. Умовно гравців можна поділити на імпульсивних марнотратів та консервативних гравців. Перші, швидше за все, витрачають ресурси одразу після їх отримання, часто на високоризиковані ставки або дорогі ігрові предмети. Вони сприяють динамічній економіці, але також можуть швидко виснажувати ресурси. На противагу цьому, консервативні гравці схильні накопичувати ресурси, витрачаючи їх обережно [4]. Розуміння такої поведінки є важливим для розробки елементів, які заохочують цих гравців до більш активної участі в економіці гри. Використання цих знань дозволяє дизайнерам адаптувати гру до різних архетипів гравців. Для азартних гравців гра може пропонувати столи з високими ставками або ексклюзивні події. Для більш обережних гравців гра може пропонувати постійні винагороди з низьким рівнем ризику, щоб утримати їх у грі.

Така кастомізація підвищує залученість гравців і забезпечує різноманітну, але водночас збалансовану спільноту гравців, що є життєво важливим для процвітання ігрової економіки.

1.2. Особливості побудови ігрових економік залежно від жанру гри

Відеоігрові економіки – це складні системи, які вимагають від розробників глибокого розуміння економічних принципів. Вони відрізняються в залежності від жанру гри, і це впливає на їх побудову та балансування.

Економіка в стратегічних іграх є одним з ключових елементів геймплея. Гравці повинні ефективно збирати та використовувати ресурси, щоб розвивати свою базу, наймати юнітів та досліджувати нові технології. Залежно від гри, це може включати різні типи ресурсів, такі як їжа, дерево, золото, нафта тощо.

Балансування економіки в стратегічних іграх є важливим завданням для розробників. Вони повинні забезпечити, що жоден ресурс не є надто цінним або неважливим. Це може бути складним, оскільки розробники повинні враховувати різні стратегії, які можуть використовувати гравці, та забезпечити, щоб жодна стратегія не була надто сильною або слабкою.

Різноманіття жанрових напрямів ігор визначає й особливі підходи до побудови й балансування економік. RPG (Role-Playing Games) та MMORPG (Massively Multiplayer Online Role-Playing Games) – це жанри ігор, де гравці виконують ролі персонажів в уявних світах. Економіка в RPG і MMORPG часто базується на системі торгівлі предметами. Гравці можуть заробляти валюту в грі, виконуючи завдання, перемагаючи ворогів або продаваючи предмети, які вони знайшли або створили. Ця валюта потім може бути використана для покупки нового обладнання, зілля, заклинань та інших предметів, які поліпшують здібності персонажа або дозволяють гравцям краще пристосуватися до світу гри. Справедливість та збалансованість економіки таких ігор включає в себе визначення вартості предметів, їх рідкості та впливу на гру. Цей процес може бути складним, оскільки він вимагає врахування багатьох факторів, таких як різні класи персонажів, рівні складності та стилі гри.

Шутери – це жанр відеоігор, який зосереджений на бойових діях, де головним елементом є стрільба від першої або третьої особи. Економіка в шутерах може бути менш вираженою, ніж в інших жанрах. Вона може включати систему прогресії персонажа, де гравці можуть витратити валюту в грі на

покращення своїх здібностей або зброї. Також може бути внутрішня валюта для покупки косметичних предметів.

Балансування в шутерах також є важливим. Розробники повинні забезпечити, що всі види зброї та здібності є збалансованими, щоб жодна з них не була надто сильною або слабкою. Це важливо для забезпечення справедливого та конкурентного ігроладу. В цілому, шутери вимагають від гравців швидкого мислення, реакції та стратегії. Розробники, зі свого боку, повинні створювати збалансовані механіки гри, щоб забезпечити цікавий та захоплюючий геймплей.

Мобільні ігри – це ігри, розроблені для гри на мобільних пристроях, таких як смартфони або планшети. Вони відрізняються від ігор на ПК або консолях за своєю портативністю, оптимізацією під сенсорні екрани та часто короткими сеансами гри. Економіка в мобільних іграх переважно використовує модель «freemium» або «free-to-play», де основна гра є безкоштовною, але гравці можуть купувати додатковий контент або переваги за реальні гроші. Це може включати все, від косметичних предметів до «швидкого просування» – можливості пропустити час очікування для певних дій.

Мобільні ігри, такі як «Clash of Clans» або «Candy Crush», часто використовують економіку, засновану на часі. Гравці можуть чекати певний час, щоб отримати ресурси, або купувати їх за реальні гроші.

1.3. Сучасний стан розроблення питання балансування ігрових економік

Добре збалансовані ігри зазвичай мають більшу довговічність, оскільки гравці продовжують повертатися, щоб вдосконалити свої навички та стратегії. Звідси, баланс у грі впливає на загальний досвід гравця та може значно вплинути на успіх або невдачу гри. Розробники витрачають значний час та ресурси на тестування та балансування своїх ігор, щоб забезпечити найкращий можливий досвід для гравців.

Сучасний стан розроблення питання балансування ігрових економік включає в себе декілька ключових аспектів. Розробники використовують різні

методології для балансування ігрових процесів, що може включати використання математичних моделей, аналізу даних гравців, тестування гри та ітеративного проектування. Штучний інтелект (ШІ) стає все більш важливим у балансуванні ігрових економік, оскільки може бути використаний для моделювання поведінки гравців, прогнозування тенденцій в економіці гри та автоматичного балансування [3].

Складні економічні моделі використовуються для балансування ігрових економік, включаючи моделювання взаємодії між різними ресурсами, механіками гри та поведінкою гравців. Дані про поведінку гравців інформують процес балансування, включаючи аналіз того, як гравці використовують ресурси, як вони реагують на зміни в економіці гри та як взаємодіють з іншими гравцями.

Сучасні ігрові технології дозволяють розробникам створювати все більш складні і динамічні ігрові економіки, використовуючи обчислювальну потужність для моделювання складних економічних систем, а також новітні технології, такі як штучний інтелект та машинне навчання, для балансування ігрових економік. Ці тенденції вказують на те, що балансування ігрових економік продовжує розвиватися і стає все більш складним і вишуканим процесом. Розробники продовжують шукати нові та інноваційні способи балансування своїх ігор, щоб забезпечити цікавий та захоплюючий досвід для гравців.

Одним з прикладів є економічна модель (рис. 1.1), розроблена на основі досвіду індустрії та практичного застосування в іграх NetEase Games. Ця модель включає чотиришарову структуру, де верхній шар відображає економічну структуру, що визначає сутність поточної економічної системи. Економіка функціонує як ринкова система, але зовнішні впливи (як людські, так і штучні) можуть змінювати її на планову економіку. Гравці, як незалежні економічні суб'єкти, беруть участь у різноманітних економічних активностях, що впливають на економічні ресурси та забезпечують їх циркуляцію в системі [11].

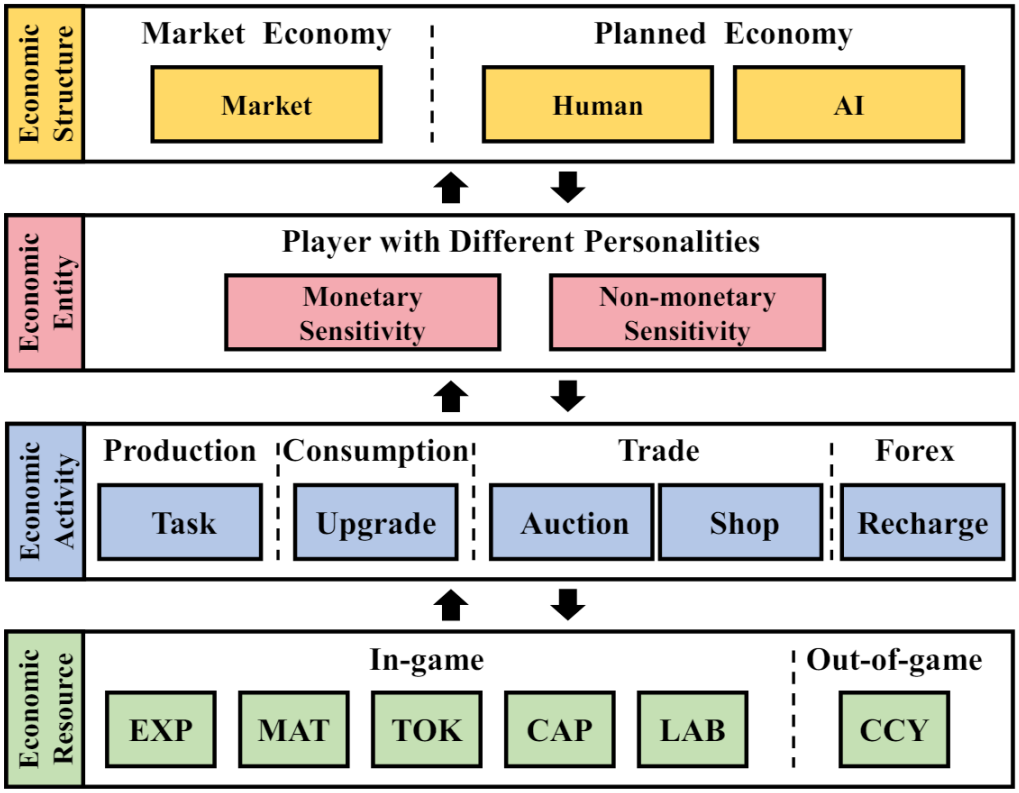

Рисунок 1.1 – Економічна концептуальна модель

Для забезпечення реалістичності та достовірності економічних симуляцій, важливо, щоб усі агенти, включаючи гравців і розробників, навчалися разом. Розробник, змінюючи налаштування P2W (Pay-to-Win), впливає на поведінку гравців, яка в свою чергу змінює очікувану прибутковість і рівність цих налаштувань. Такий спільний процес навчання створює нестабільний ландшафт винагород як для гравців, так і для розробника. Для вирішення цього питання, була запропонована ієрархічна структура (рис. 1.2), що використовує комбінацію підходів підкріпленого навчання (RL) у внутрішньому і зовнішньому циклах. У внутрішньому циклі, гравці за допомогою мультиагентного підкріпленого навчання (MARL) набувають досвіду, активно беручи участь в економіці MMO (Масова багатокористувацька онлайн-гра) та взаємодіючи з іншими гравцями. У зовнішньому циклі, розробник на основі RL налаштовує політики P2W для оптимізації прибутковості та рівності, враховуючи найкращі відповіді гравців на ці налаштування.

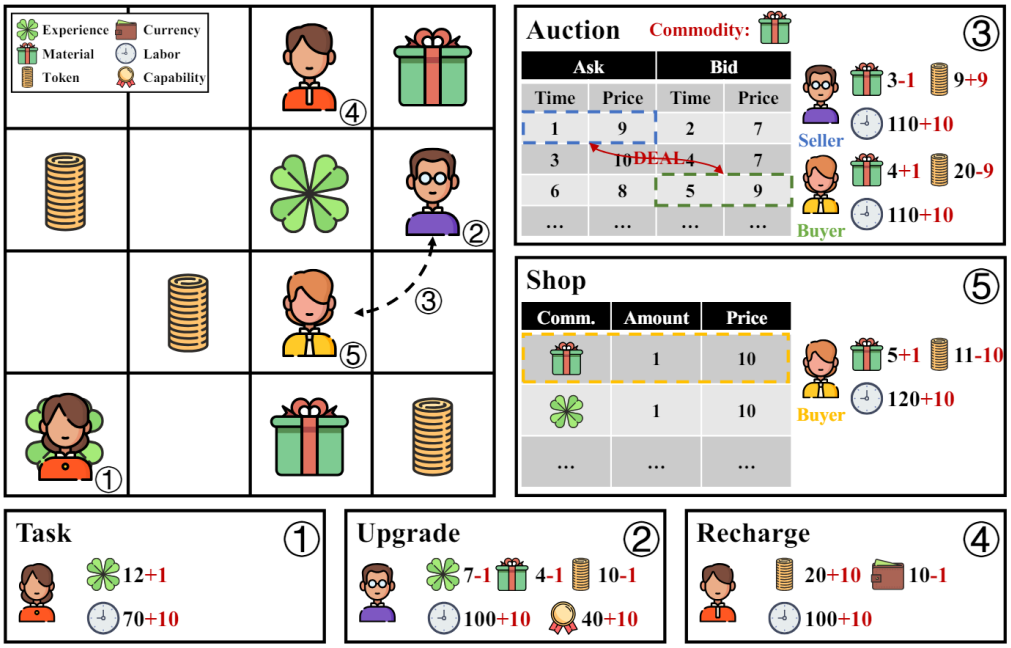

Рисунок 1.2 – Демонстрація економічного моделювання для MMO

Підхід також включає побудову симуляційного середовища, що дозволяє економістам MMO моделювати та оптимізувати економічні стратегії, проводити тестування та впроваджувати перевірені рішення у виробничому середовищі. Таким чином, сучасні методи балансування ігрових економік включають застосування передових технологій, таких як штучний інтелект і підкріплене навчання, що дозволяє створювати ефективні та адаптивні економічні системи, які забезпечують цікавий та збалансований ігровий процес [11].

РОЗДІЛ 2
ПРОЄКТУВАННЯ ІГРОВОЇ ЕКОНОМІКИ

2.1. Аналіз вимог до програмного забезпечення

Відеоігрова економіка майбутньої гри повинна містити декілька ключових компонентів: джерела ресурсів та способи їх витрачання. Основними ресурсами є золото і їжа, які використовуються для будівництва та утримання об’єктів у грі. Стартова кількість золота та їжі встановлюється на початку гри і відображається у відповідних показниках.

Виробництво ресурсів здійснюється золотими шахтами та фермами, які забезпечують постійний приплив золота та їжі. Захисні вежі споживають ці ресурси для утримання і автоматично витрачають їх з певною періодичністю. Якщо ресурси закінчуються, вежі не можуть утримуватися, що призводить до автоматичної поразки.

Також економічна модель має включати механіку покращень для всіх будівель. Покращення для ферм та шахт повинні впливати на кількість ресурсів, виробленими цими будівлями. У свою чергу, покращення для захисних веж мають поліпшити їх швидкість стрільби та збільшити споживання ресурсів.

Взаємодія елементів ігрової економіки забезпечується балансом між кількістю побудованих золотих шахт, ферм і захисних веж. Ефективна взаємодія між цими елементами дозволяє гравцю планувати свої дії та забезпечувати стабільний розвиток економіки гри. Гравець має уважно стежити за змінами в ресурсах і вчасно реагувати на потреби економіки, щоб уникнути дефіциту і зберегти оборонні можливості.

2.2. Нотація Machinations для проєктування ігрової економіки

Machinations – це браузерний інструмент для проєктування та балансування ігрових систем. Він призначений для моделювання діяльності, взаємодії та комунікації між частинами внутрішньої економіки гри [7].

В економічній системі гри домінує потік ресурсів. Для моделювання внутрішньої економіки гри діаграми Machinations використовують кілька типів

вузлів, які збирають і розподіляють ресурси. Ресурсні зв’язки визначають, як ресурси переміщуються між елементами, а зв’язки стану визначають, як поточний розподіл ресурсів змінює інші елементи на діаграмі. Разом ці елементи формують основне ядро діаграми Machinations [9].

Пул (Pool) – це місце на діаграмі, де збираються ресурси. Пули зображуються у вигляді відкритих кружечків, а ресурси, які зберігаються в пулі, зображуються у вигляді менших кружечків, які складаються на них. Якщо в пулі занадто багато ресурсів, щоб показати їх у вигляді стеків, інструмент показує замість них число [7]. Відображення пулів на діаграмах Machinations продемонстровано на рис. 2.1.

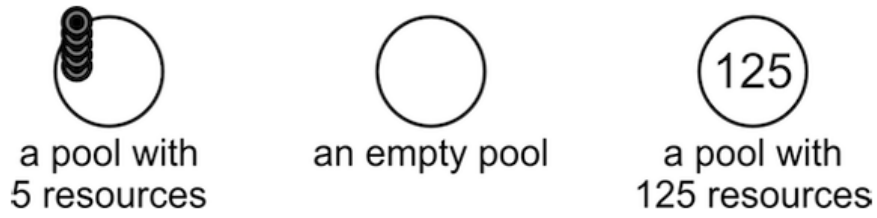

Рисунок 2.1 – Вигляд пула на діаграмі

Пули використовуються для моделювання сутностей. Наприклад, якщо у вас є ресурс «гроші» і сутність «банківський рахунок гравця», ви можете використати пул для моделювання банківського рахунку.

Найпростіший тип зв’язку – це зв’язок ресурсів (resource connection), який передає ресурси від одного вузла до іншого. Вони зображуються суцільними стрілками, що з’єднують вузли діаграми (рис. 2.2). Ресурсні з’єднання можуть передавати ресурси з різною швидкістю. Мітка біля з’єднання ресурсів показує, скільки ресурсів може переміщатися вздовж з’єднання за один часовий крок. Якщо ресурсне з’єднання не має мітки, його швидкість вважається рівною 1 [7].

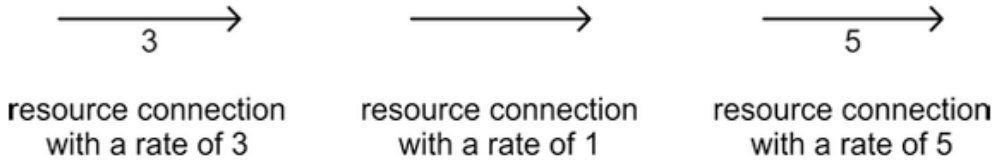

Рисунок 2.2 – Вигляд з’єднання на діаграмі

Пул може перебувати в одному з чотирьох різних режимів активації, як показано на рис. 2.3.

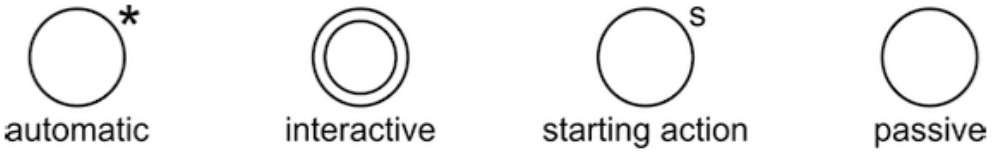

Рисунок 2.3 – Вигляд різних станів пула на діаграмі

Коли пул запускається, він намагається витягнути ресурси з усіх підключених до нього входів. Кількість ресурсів, які він витягує, визначається швидкістю з’єднання окремих вхідних ресурсів. Крім того, пул можна налаштувати в режимі виштовхування. При цьому, коли пул спрацьовує, він виштовхує ресурси через свої вихідні з’єднання зі швидкістю потоку вихідного з’єднання ресурсів. Різні типи відображення з’єднань наведено на рис. 2.4.

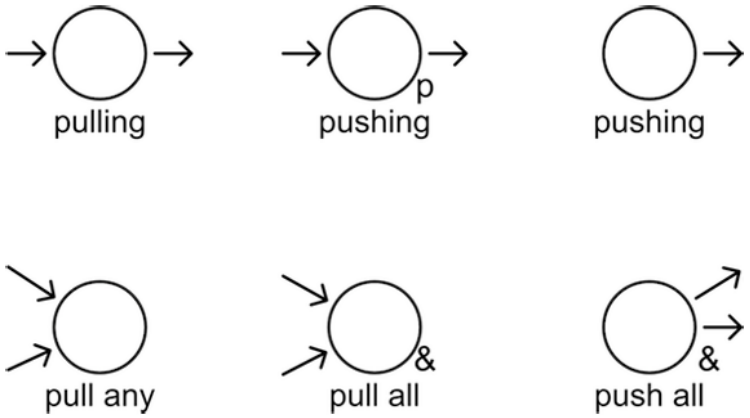

Рисунок 2.4 – Типи з’єднань пула на діаграмі

Використовуючи пули та з’єднання ресурсів, ми можемо побудувати простий пісочний годинник (рис. 2.5). У цьому випадку два пули з’єднані одним ресурсним з’єднанням. Верхній пул (A) є пасивним і містить п’ять ресурсів, тоді як нижній пул (B) є автоматичним і запускається без будь-яких ресурсів. Після кожної ітерації B буде забирати по одному ресурсу з A, поки всі ресурси не перейдуть з A до B. Після цього стан цієї діаграми більше не змінюється.

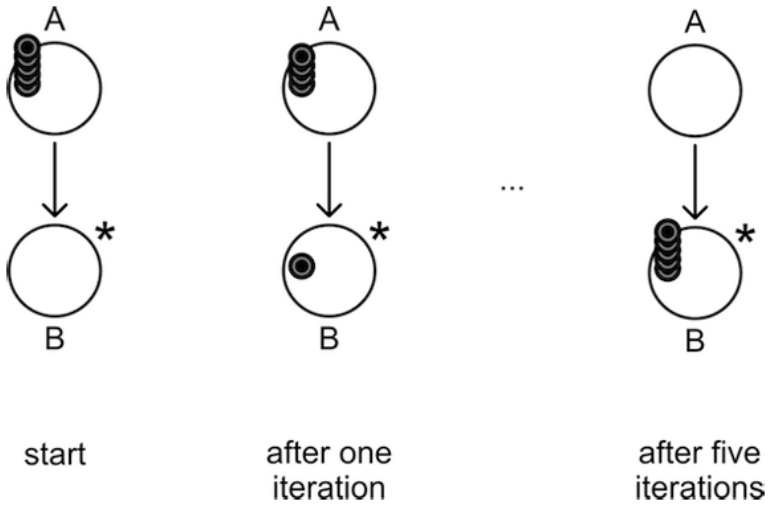

Рисунок 2.5 – Приклад пісочного годинника

Стан діаграми – це поточний розподіл ресурсів між її вузлами. Коли ресурси переміщуються з одного місця в інше, стан змінюється. У фреймворку Machinations ви можете використовувати зміни стану для зміни швидкості потоку з'єднань ресурсів. Крім того, ви можете запускати вузли, активувати або деактивувати їх у відповідь на зміни в розподілі ресурсів. Щоб зробити це можливим, Machinations пропонує другий клас зв'язків, які називаються зв'язками станів. Вони показують, як зміни поточного стану вузла (кількість ресурсів у ньому) впливають на щось інше на діаграмі.

Зв'язки станів зображуються пунктирними стрілками, що ведуть від керуючого вузла до цілі, якою може бути як вузол, так і ресурсний зв'язок, або, рідше, інший зв'язок стану. Мітки на з'єднанні стану вказують на те, як воно змінює ціль.

Модифікатор мітки вказує, як зміни стану в вузлі-джерелі змінюють поточне значення цільової мітки на поточному часовому кроці, як зазначено у власній мітці з'єднання стану (рис. 2.6). Нове значення набуває чинності на наступному часовому кроці. Величина зміни у початковому вузлі множиться на власну мітку множника мітки.

Отже, якщо модифікатор мітки має значення +3, а вихідний вузол збільшується на 2, то цільова мітка збільшиться на 6 на наступному часовому кроці (вона додасть 3 двічі, по одному разу на кожну зміну у вихідному вузлі).

Однак, якщо модифікатор мітки +3 і початкова вершина зменшується на 2, то цільова мітка зменшиться на 6 [7].

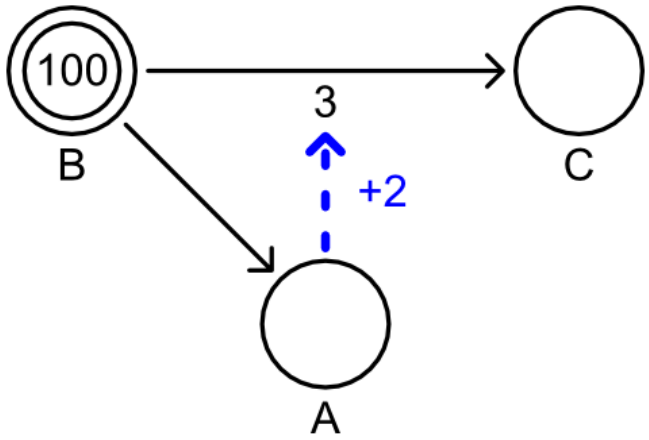

Рисунок 2.6 – Вигляд модифікатора мітки на діаграмі

Синя пунктирна лінія на рис. 2.6 є модифікатором мітки. Мітка завжди починається з символу плюс або мінус. Наприклад, на рисунку вище кожен ресурс, доданий до пулу A, додає 2 до значення потоку ресурсів між пулами B і C. Отже, при першій активації B одиниця ресурсу переходить до A і три одиниці ресурсу переходять до C. При другій активації одиниця ресурсу все ще переходить до A, але тепер п’ять одиниць ресурсу переходять до C.

Якщо ви хочете додати в свою гру випадковість або можливість вибору, варто використовувати гейти (Gate). Вони можуть розподіляти ресурси між різними пулами. Звідси, ви можете використовувати їх, наприклад, для імітації випадання предметів. Можна надавати гейтам детерміновані або випадкові значення того, як вони мають розподіляти ресурси. Тобто ви можете створити систему, яка викидає 2 синіх предмети і один зелений, коли ви вбиваєте ворога. Це буде заздалегідь визначено. Випадковим може бути те, що у вас є 66% шанс отримати синій предмет і 33% шанс отримати зелений. Гейти також можуть мати кілька інших варіантів використання.

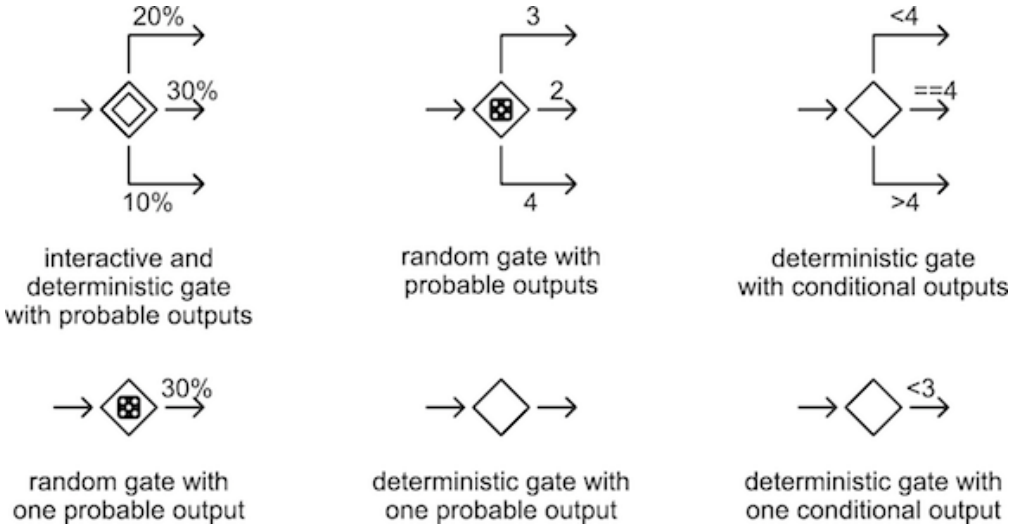

Рисунок 2.7 – Види компоненту гейт на діаграмі

Джерела (Source) – це вузли, які створюють ресурси. Вони зображуються у вигляді трикутника, спрямованого вгору. Як і будь-який інший вузол на діаграмі, джерела можуть бути автоматичними (за замовчуванням), інтерактивними, пасивними або активуватися один раз перед початком діаграми (рис. 2.8). Прикладом автоматичного джерела є постійна регенерація захисних щитів зоряного винищувача гравця в Star Wars: X-Wing Alliance. Дії з побудови армій у Risk можна змоделювати як інтерактивне джерело армій, а проходження гри Go у Monopoly – як пасивне джерело грошей, яке запускається ігровою подією [7]. Швидкість, з якою джерело виробляє ресурси, є фундаментальною властивістю джерела і позначається швидкістю потоку його результатів.

Рисунок 2.8 – Вигляд джерела у Machinations

Поглиначі (Drain) – це вузли, які споживають ресурси; ресурс, який потрапляє туди, зникає безповоротно. У фреймворку Machinations є спеціальний вузол поглинача, представлений у вигляді трикутника, спрямованого донизу (рис. 2.9). Швидкість поглинача визначається швидкістю потоку його вхідного ресурсного з’єднання. Деякі поглиначі споживають ресурси з постійною швидкістю, тоді як інші споживають ресурси з випадковою швидкістю або через

випадкові проміжки часу. Ви також можете змусити його споживати всі ресурси, до яких він приєднаний, позначивши його з’єднання з ресурсом «усі» [7].

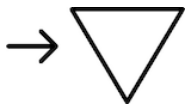

Рисунок 2.9 – Вигляд поглинача у Machinations

Ми також можемо використовувати блок під назвою конвертор (Converter). Вони можуть, як випливає з назви, перетворювати одні ресурси в інші. Якщо ви отримали валюту від видобутку золота, можливо, ви хочете перетворити його на монети чи злитки. Тоді вам знадобляться конвертери (рис. 2.10) [6].

Рис. 2.10. Вигляд конвертора у Machinations

Трейдери (Trader) – це вузли, які змушують ресурси змінювати власника, коли їх звільняють: два гравці можуть використовувати трейдера для обміну ресурсами. На діаграмах трейдер зображується вертикальною лінією над двома трикутниками, які спрямовані вліво і вправо (рис. 2.11). Їх використовують, коли певна кількість ресурсів одного типу обмінюється на певну кількість ресурсів іншого типу (а не перетворюється на них). Це ідеально підходить для будь-якої ситуації, що нагадує купівлю: продавець отримує гроші, а покупець – товари у визначеній пропорції (ціна). Якщо або продавець, або покупець не мають необхідних ресурсів, торгівля не може відбутися. Хорошим прикладом є гра Fallout 3, в якій запаси всіх торговців обмежені.

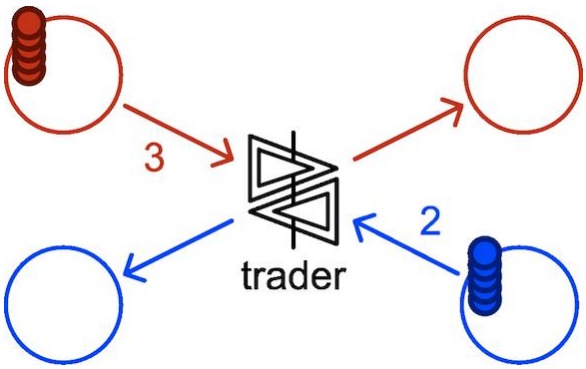

Рисунок 2.11 – Вигляд трейдера у Machinations

Останнім стандартним блоком є затримка (Delay). За допомогою затримки можна сповільнити транспортування ресурсів. Також можливо використовувати її для імітації часу будівництва, часу в дорозі або часу виробництва. Наприклад, процес перетворення золота в монети може зайняти певний час, тому можна використовувати затримку для цього [6].

Рисунок 2.12 – Вигляд елементу затримки

Machinations дозволяє моделювати та імітувати динамічні системи на довільних рівнях абстракції. Наприклад, основний цикл позитивного зворотного зв’язку в грі Монополія показано на рис. 2.13.

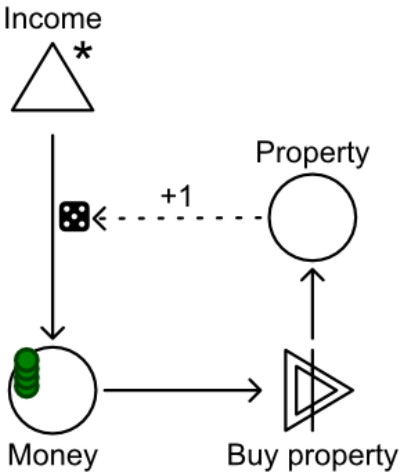

Рисунок 2.13 – Приклад циклу гри Монополія побудований у Machinations

2.3. Модель ігрової економіки в Machinations

Перед створенням діаграми в Machinations необхідно визначити основні елементи ігрової економіки: золото, їжу, золоті шахти, ферми та захисні вежі. Золото є початковим ресурсом, що використовується для будівництва золотих шахт, ферм і захисних веж.

Діаграма створюється шляхом додавання вузлів та зв’язків між ними, що відображають потоки ресурсів та взаємодії. Для ресурсів використовуються два пули ресурсів: один для золота, інший для їжі. У цих пулах вказується початкова кількість ресурсів.

Для виробництва ресурсів додаються джерела ресурсів для золотих шахт та ферм, що генерують ресурси у відповідні пули. Для споживання ресурсів додаються поглиначі ресурсів для захисних веж, які споживають золото та їжу з пулів для утримання. Для побудови об’єктів використовуються конвертери ресурсів, де вхідними ресурсами є витрати золота, а вихідними ресурсами – кількість побудованих об’єктів.

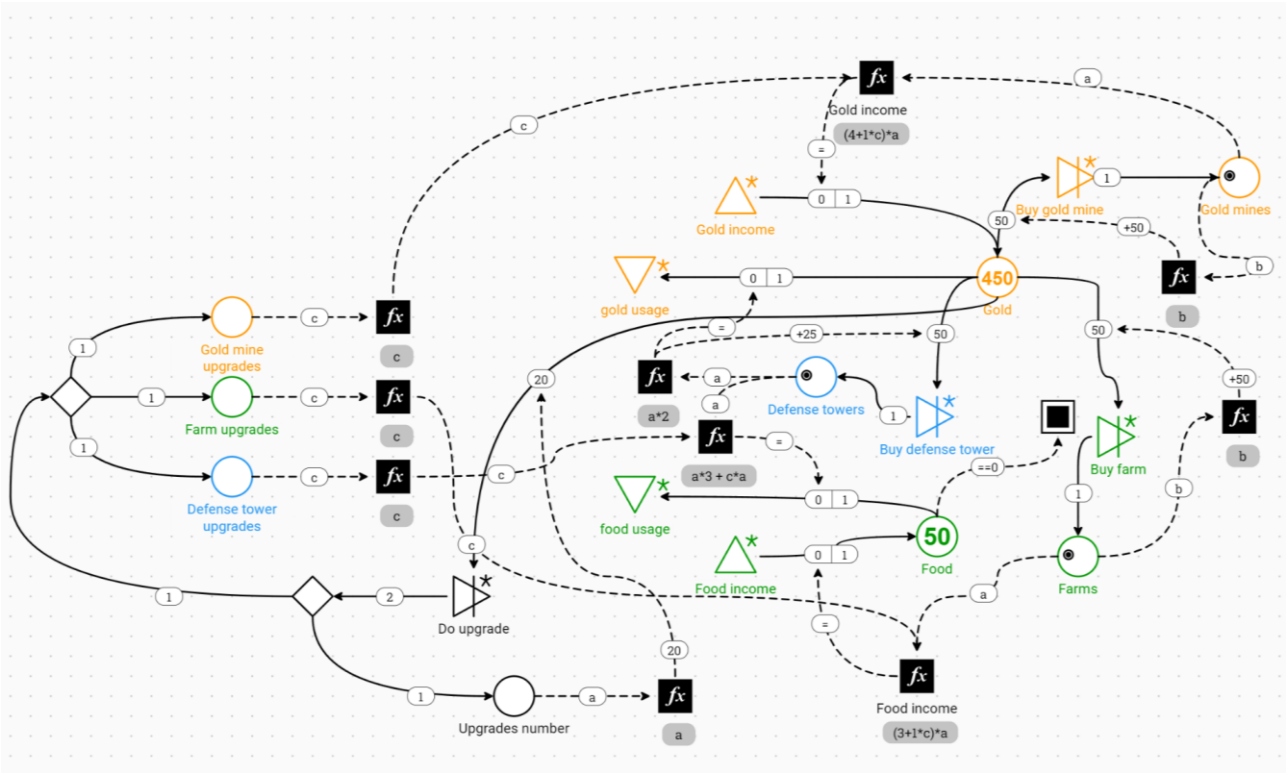

Рисунок 2.14 – Побудована модель економіки у Machinations

Діаграма відображає взаємодію між різними компонентами ігрової економіки. На початку гри маємо певну кількість золота та їжі в пулах. Золоті шахти та ферми генерують золото та їжу відповідно, додаючи ресурси до відповідних пулів з певною періодичністю.

Модель витрачає золото на побудову нових золотих шахт, ферм або захисних веж через конвертори ресурсів, а також на купівлю апгрейдів, як показано на рис. 2.15. Наприклад, якщо придбано нову ферму, кількість золота в пулі для золота зменшується, але виробництво їжі збільшується.

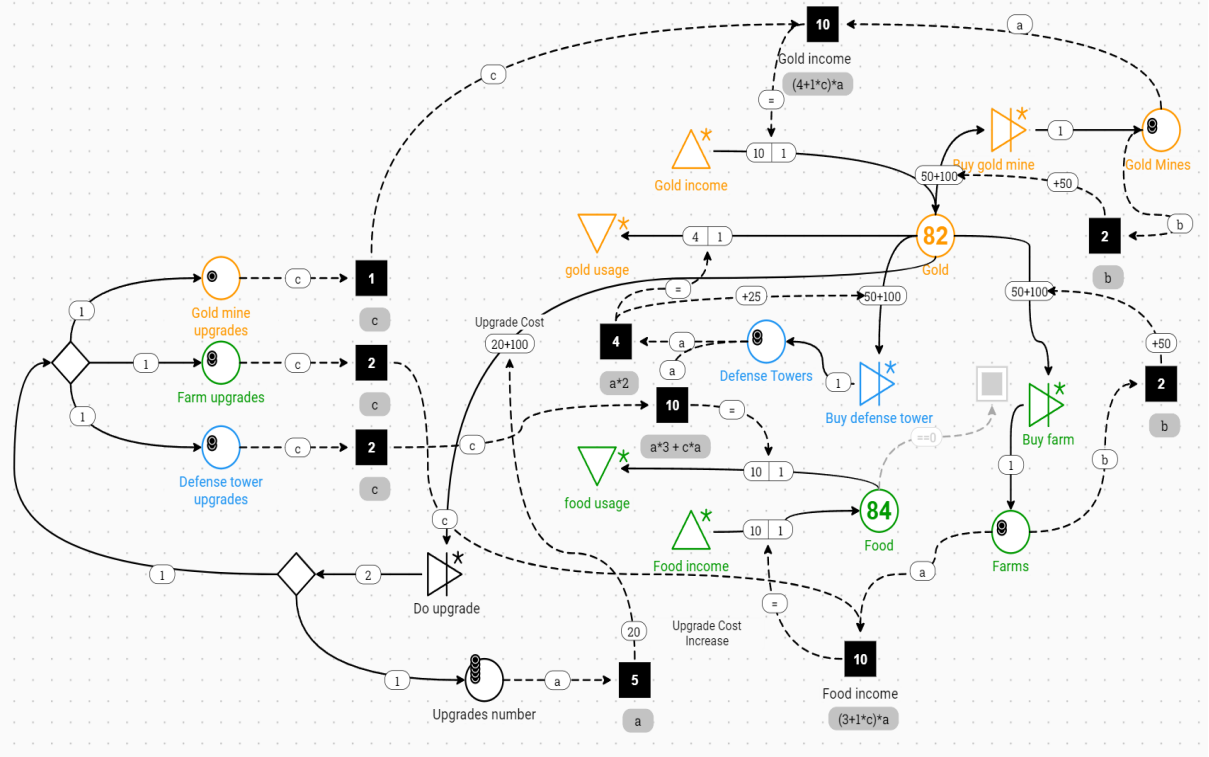

Рисунок 2.15 – Діаграма Machinations під час симуляції

Захисні вежі споживають золото та їжу для утримання. Поглиначі ресурсів для веж автоматично знімають певну кількість ресурсів з пулів з певною періодичністю. Баланс ресурсів залежить від кількості наявних покращень для будівель, кількості побудованих золотих шахт та ферм, які генерують ресурси, та від кількості захисних веж, які їх споживають. Цикл отримання золота крупним планом продемонстровано на рис. 2.16.

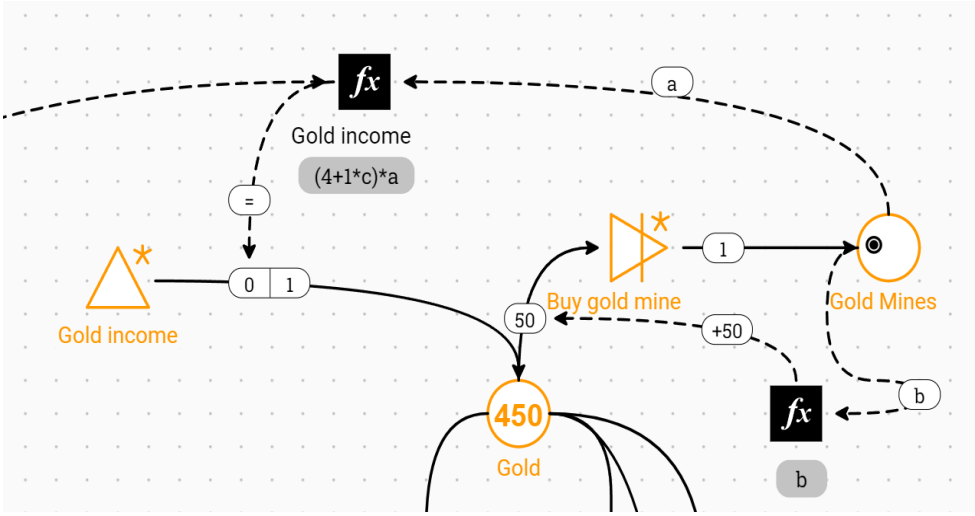

Рисунок 2.16 – Цикл отримання золота

На цьому рисунку зображена частина діаграми ігрової економіки, яка відображає цикл отримання золота, а також покупку золотих шахт. Отримання золота обчислюється за формулою $(4 + 1 \cdot c) \cdot a$, де a – кількість золотих шахт, а c – кількість апгрейдів для них. Відповідно до такого закону розподілу отримаємо динаміку зростання обсягу золота в грі, що показано на рис. 2.17.

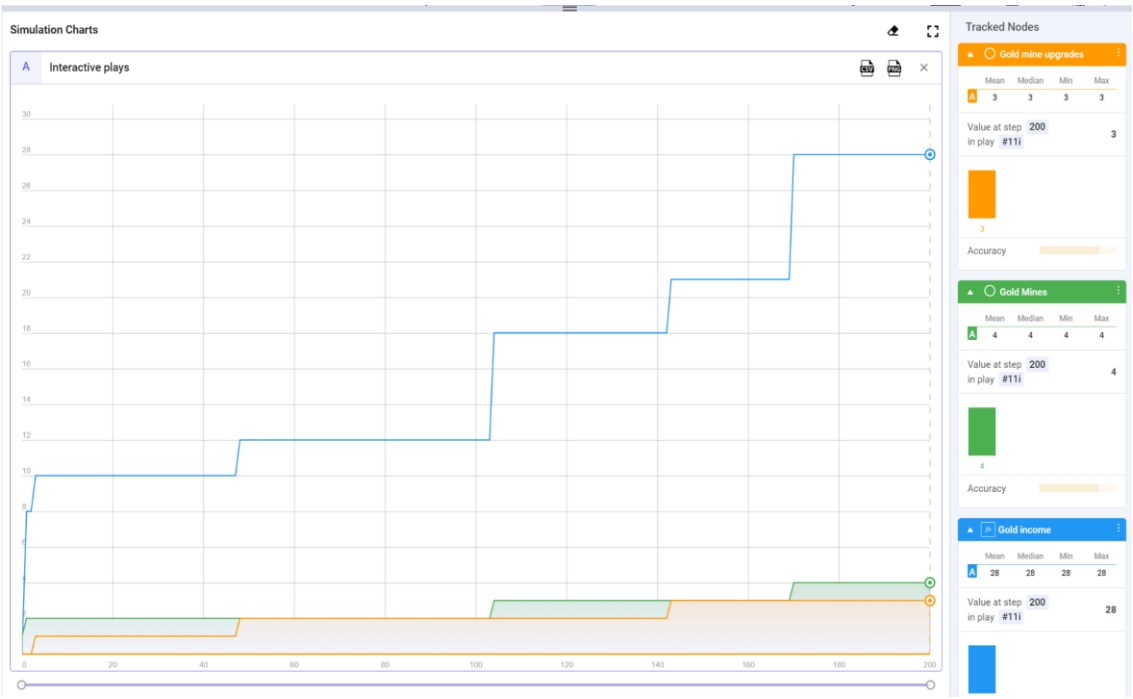

Рисунок 2.17 – Графік доходу золота

Пул золота представляє запас золота, встановленого на початку гри. Джерело отримання золота генерує певну кількість золота з певною

періодичністю і додає його до пулу. Конвертер для покупки золотих шахт дає змогу витрачати золото з пулу золота для створення нових золотих шахт. Кожного разу, коли необхідно купити шахту, певна кількість золота з пулу золота витрачається і додається до пулу шахт, який відображає кількість активних золотих шахт у грі.

Пул шахт представляє кількість побудованих золотих шахт. Кожна активна шахта збільшує кількість золота, яке генерується джерелом отримання золота. Також кожна побудована шахта збільшує ціну покупки наступної на 50 золота. Наближена частина діаграми, яка відповідає за логіку витрати ресурсів, продемонстрована на рис. 2.18.

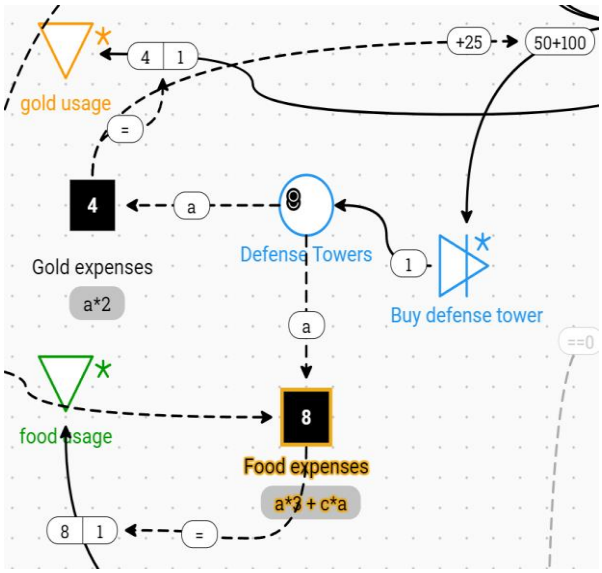

Рисунок 2.18 – Частина діаграми, яка обчислює витрати ресурсів

Аналогічно працюють пули та конвертери для ферм та захисних веж. Проте відмінністю пулу веж є те, що кількість захисних споруд у пулі впливає не на джерела, які постачають ресурси, а на поглиначі, які їх витрачають. Тобто якщо в пулі буде 2 захисні вежі, то витрати золота на їх утримання будуть дорівнювати добутку кількості веж на 2. У свою чергу, витрати їжі будуть обчислені за формулою $3a + c \cdot a$, де a – кількість веж, а c – кількість їх покращень. Відповідна динаміка витрати їжі наведена на рис. 2.19.

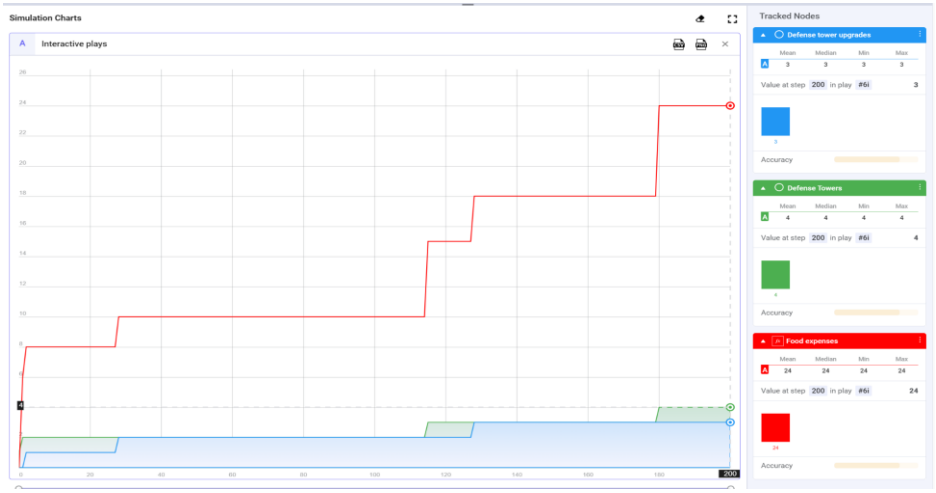

Рисунок 2.19 – Графік витрати їжі

На діаграмі ці обчислення виконуються в спеціальних компонентах під назвою регістр (Registr). Ці компоненти за допомогою зв’язків отримують необхідні значення змінних для обчислення формули і потім передають це значення до поглинача вказуючи тому скільки ресурсів потрібно використовувати. Обчислення в регістрах виконуються автоматично при зміні будь-якого параметру.

Іншою частиною діаграми є блок компонентів, що відповідають за купівлю апгрейдів для споруд. На рис. 2.20 показано збільшений блок діаграми для купівлі покращень. Тут, як і з будівлями, за допомогою регістру визначається ціна покупки, перший апгрейд коштує 20 золота, а ціна наступного збільшується на 20. Динаміка росту цін покращень наведена на рис. 2.21.

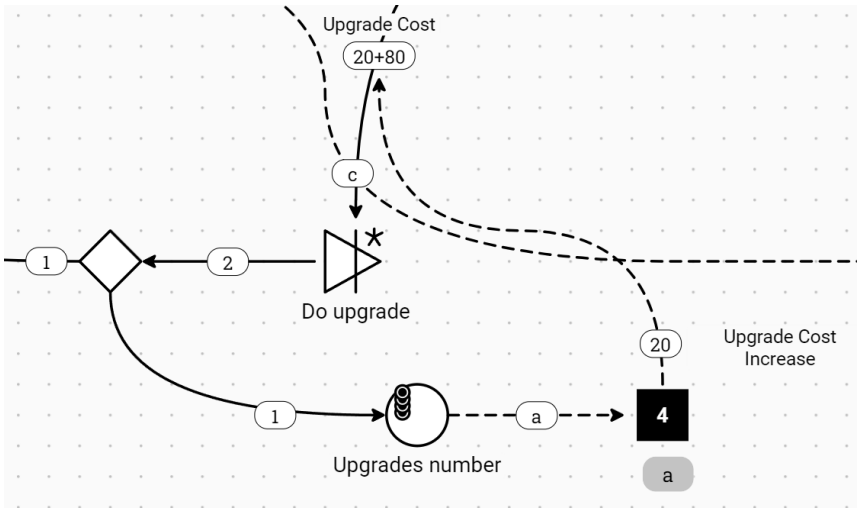

Рисунок 2.20 – Блок діаграми для купівлі покращень

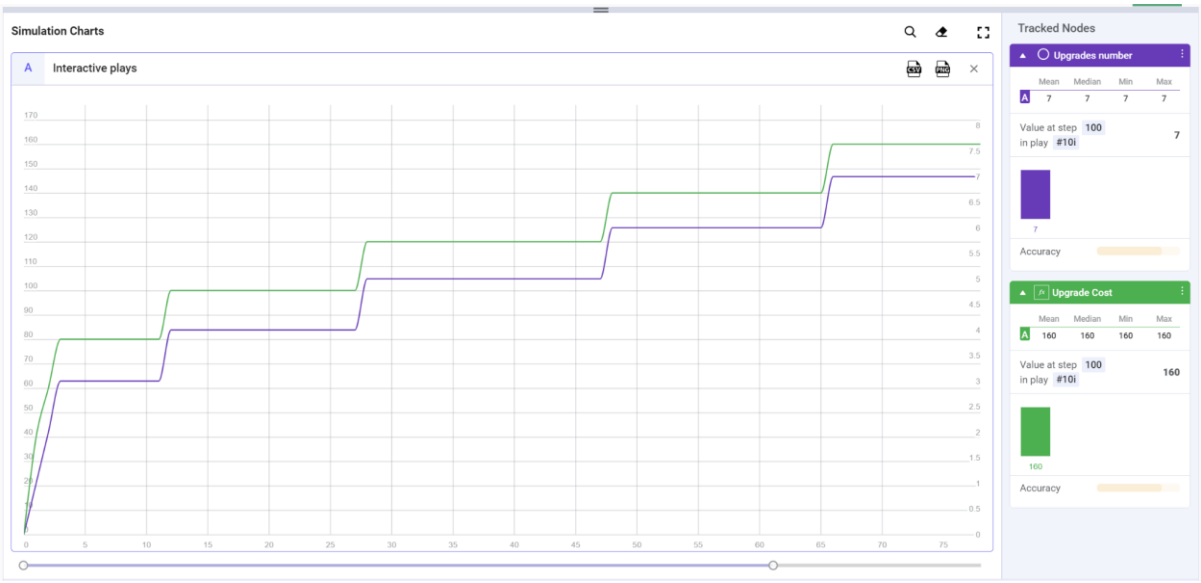

Рисунок 2.21 – Графік росту цін на покупку покращень

Логіка процесу покупки апгрейду схожа до купівлі споруд, проте конвертер перетворює золото не відразу в певний апгрейд, а створює два об’єкти і за допомогою гейта розділяє їх по одному у різні напрямки. Один об’єкт йде до пулу апгрейдів і збільшує загальне їх число на 1, а інший – до другого гейта, який визначає, в який пул апгрейдів відправити цей об’єкт. Гейти можуть містити різноманітну логіку для визначення в якому напрямку і в якій кількості відправити ресурси. Механізм розподілу покращень відображено частиною діаграми Machinations, показаною на рис. 2.22.

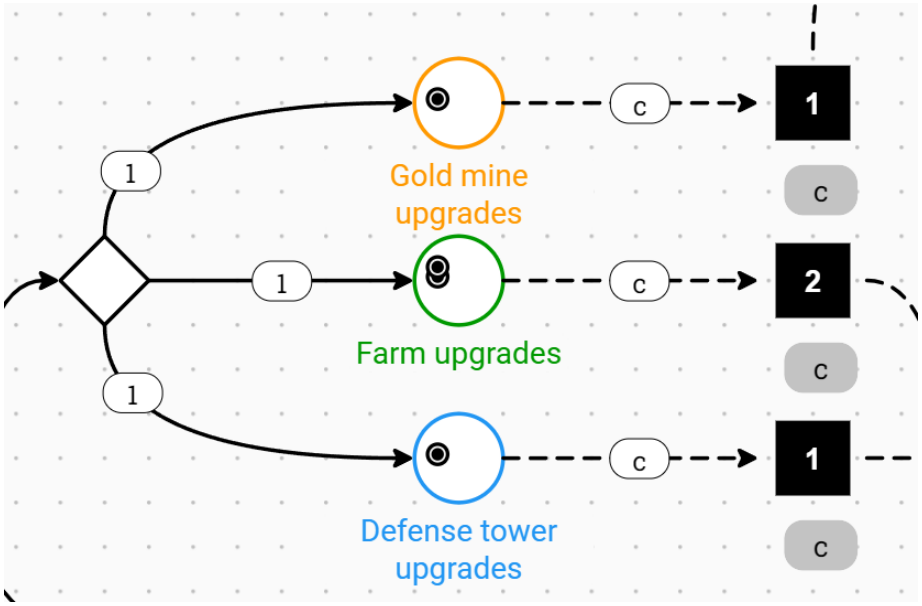

Рисунок 2.22 – Складник діаграми, що містить механізм розподілу покращень

У нашому випадку гейт налаштований таким чином, що кожний пул буде послідовно збільшуватися на 1. Тобто ресурс, відправлений до гейту буде відправлений до найменшого пулу, якщо ж вони будуть однаковими пріоритет буде спочатку у центрального пулу, а потім у нижнього. Динаміка купівлі покращень теж відстежується системою, що продемонстровано на рис. 2.23.

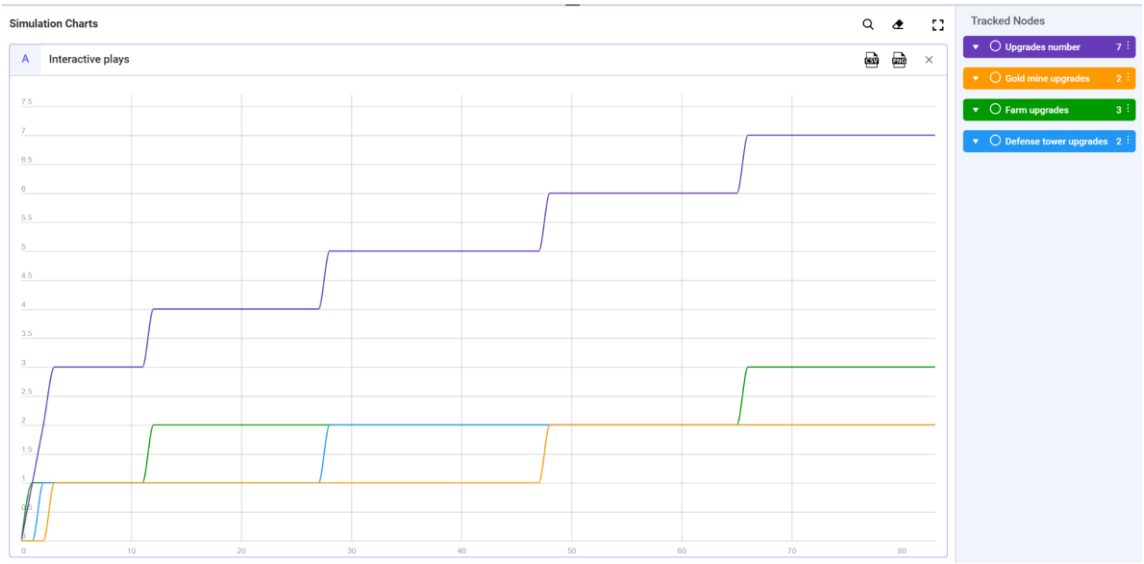

Рисунок 2.23 – Графік купівлі покращень

За результатами розгляду питань проєктування відеоігрової економіки було побудовано модель засобами середовища Machinations. Запропонована модель буде основою для демонстраційного прототипу ігрового додатка, що показуватиме життєздатність та збалансованість ігрової економіки.

РОЗДІЛ 3
ВПРОВАДЖЕННЯ МОДЕЛІ ІГРОВОЇ ЕКОНОМІКИ

3.1. Розробка програмного забезпечення

Розгортання прототипу гри вимагає декількох кроків, включаючи налаштування середовища розробки та експорт проекту в потрібний формат. Налаштування проєкту в Unity починається зі створення нового проєкту, налаштування основної сцени та додавання необхідних об’єктів, таких як Canvas для UI, спрайти для будівель та ворогів. Сформована сцена міститиме дерево об’єктів, як показано на рис. 3.1.

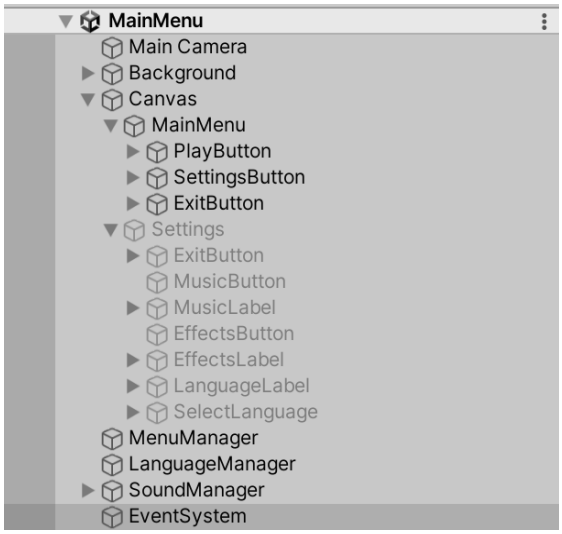

Рисунок 3.1 – Ієрархія об’єктів на сцені Unity

Налаштування середовища розробки передбачає встановлення та налаштування Unity, імпорт необхідних активів у проєкт та конфігурацію параметрів гравця та інших налаштувань проєкту, які зображені на рис. 3.2. та рис. 3.3.

Експорт проєкту включає налаштування Build Settings та Player Settings в Unity та створення виконавчого файлу гри для вибраної платформи, у нашому випадку для платформи Android.

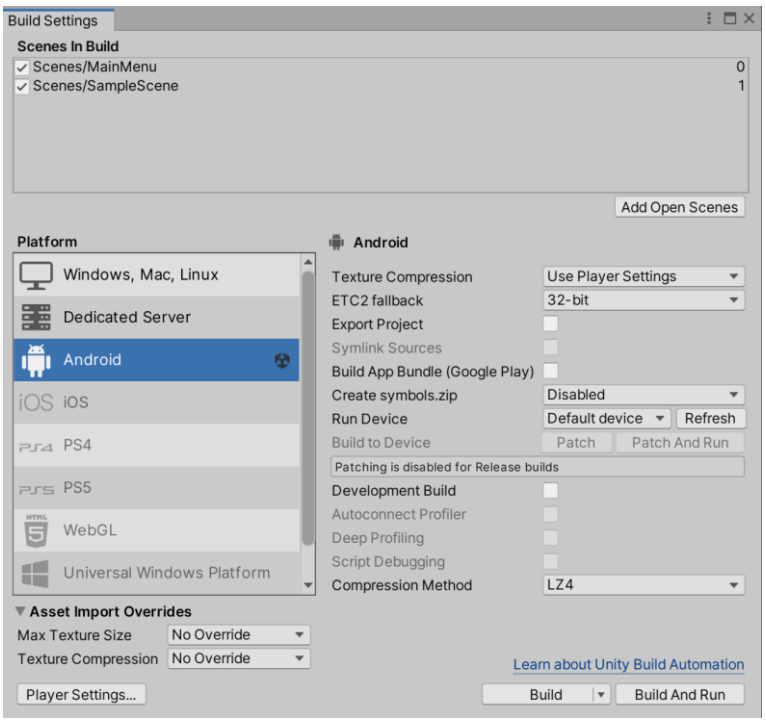

Рисунок 3.2 – Налаштування для платформи Android

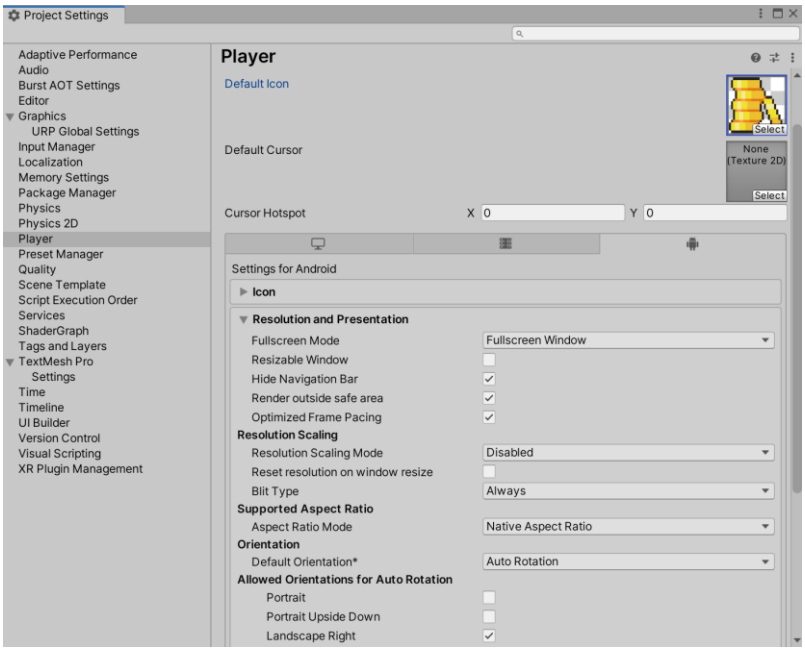

Рисунок 3.3 – Налаштування параметрів гри

Створення моделі економіки гри в інтерфейсі Machinations є ключовим етапом. Для використання Machinations UP в Unity, потрібно завантажити та імпортувати плагін за допомогою менеджера пакетів (Package Manager), як показано на рис. 3.4.

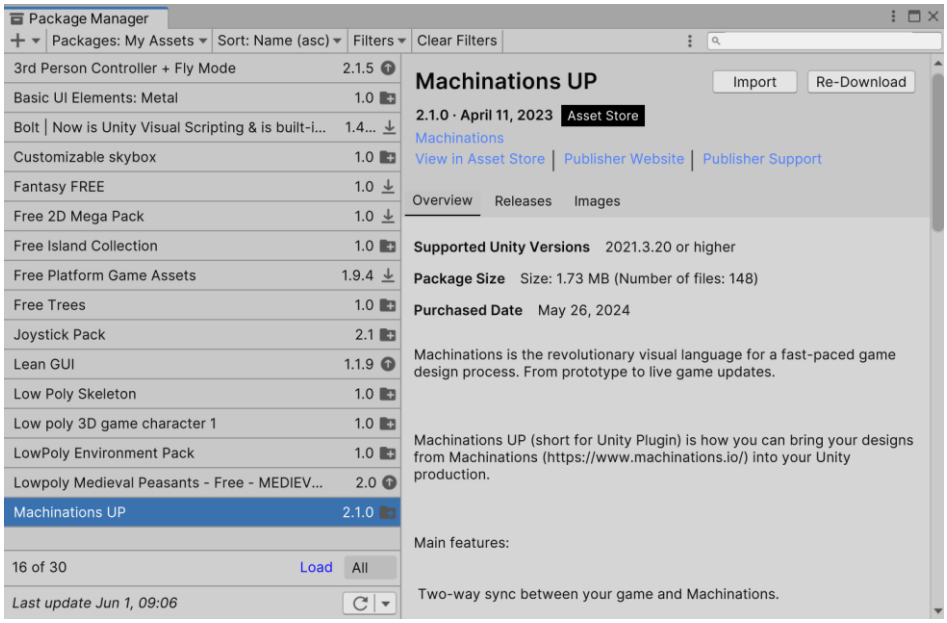

Рисунок 3.4 – Менеджер для управління пакетами

Після цього слід підключити його до свого облікового запису Machinations та імпортувати створену модель, як показано на рис. 3.5. Потім можна використовувати її в проєкті для аналізу та тестування економічної системи гри.

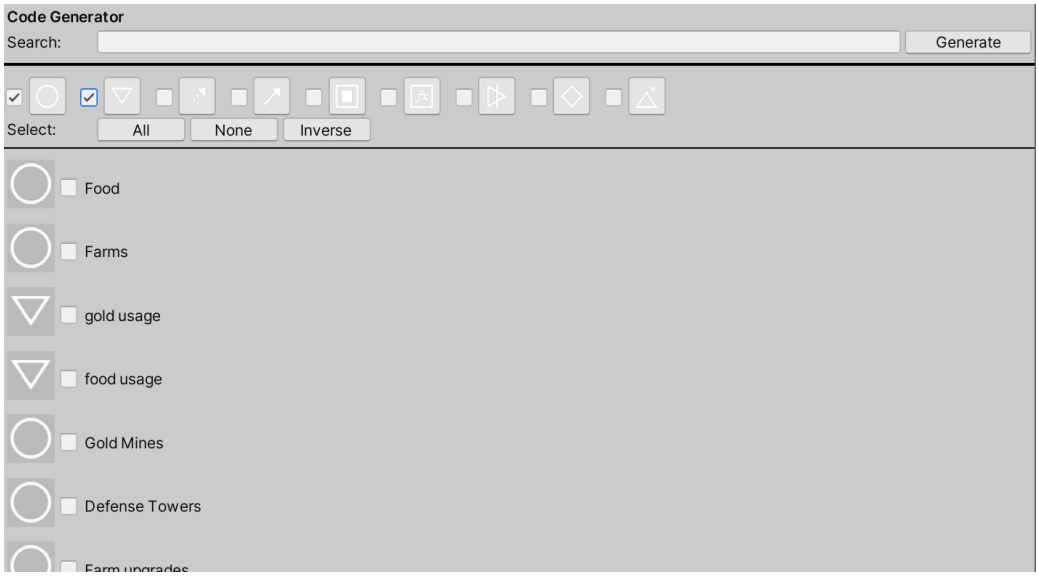

Рисунок 3.5 – Вікно вибору компонентів діаграми для імпорту

Імпорт моделі працює таким чином: на панелі керування плагіну Machinations UP потрібно обрати компоненти, дані яких необхідно імпортувати в гру. Далі після натиснення на кнопку генерується скрипт для Unity мовою програмування C#. Цей скрипт дає змогу створити спеціальний контейнер –

Scriptable Object (рис. 3.6), який використовує ігровий рушій для зручного зберігання даних.

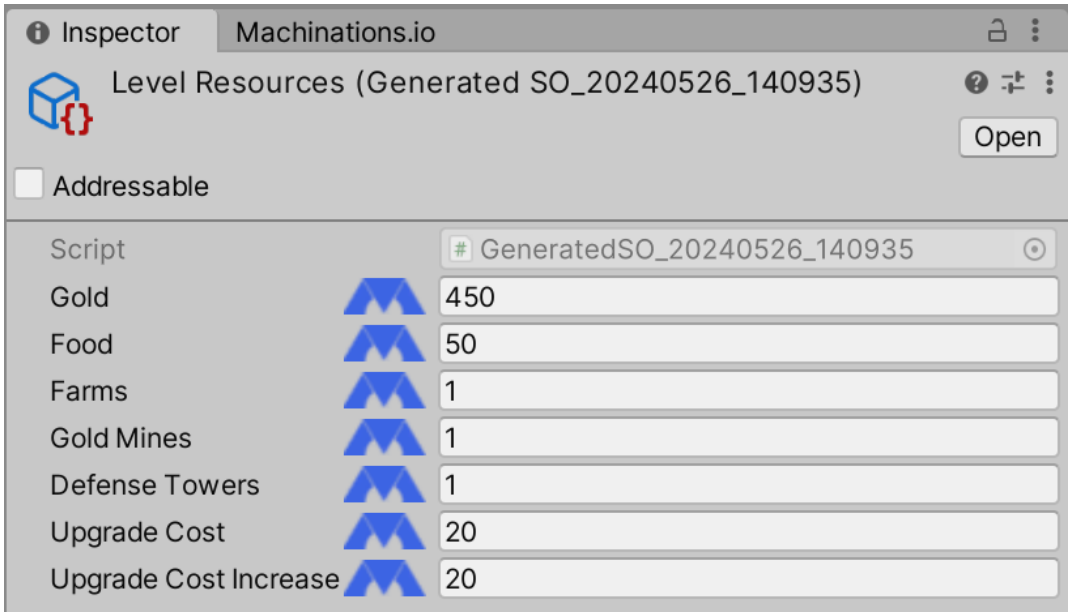

Рисунок 3.6 – Вигляд Scriptable Object у Unity

Створивши цей об’єкт у редакторі Unity, ми отримуємо доступ до даних діаграми, яку імпортували. Ці дані синхронізуються з діаграмою Machinations. При будь-якій зміні діаграми дані будуть автоматично змінюватися під час запуску гри в редакторі. Аналогічно при зміні даних в Scriptable Object діаграма буде оновлюватися. Це дає змогу зручно вносити зміни в гру без необхідності втручання в код.

3.2. Побудова демонстраційного прототипу гри

Для демонстраційного прототипу гри було обрано ігровий рушій Unity. Використовуючи мову програмування C# та архітектуру MVC (Model-View-Controller), було розроблено прототип, що реалізує економічну модель, описану в діаграмі Machinations.

Модель відповідає за дані та логіку ворогів і стріл. Вороги мають характеристики, такі як здоров’я та швидкість (лістинг 3.1), а стріли – шкоду та швидкість польоту (лістинг 3.2).

Лістинг 3.1 - Клас EnemyModel

```
public class EnemyModel {
    public Action OnDied;

    private int _health = 5;
    private float _moveSpeed = 1.5f;
    private int _damage = 1;
    private bool _isAttack = false;

    public int Health { get => _health; }
    public float MoveSpeed { get => _moveSpeed; }
    public int Damage { get => _damage; }
    public bool IsRun { get; set; }
    public bool IsAttack { get => _isAttack;
        set {
            _isAttack = value;
        }
    }

    public void TakeDamage(int damage) {
        _health -= damage;
        if (_health <= 0) OnDied();
    }
}
```

Клас EnemyModel визначає основні властивості ворогів, такі як здоров’я (_health), швидкість руху (_moveSpeed), шкода (_damage) та стан атаки (_isAttack). Метод TakeDamage() зменшує здоров’я ворога при отриманні шкоди і викликає подію OnDied(), якщо здоров’я знижується до нуля.

Лістинг 3.2 – Клас ArrowModel

```
public class ArrowModel {
    private float _moveSpeed = 15f;
    private int _damage = 1;
    private  float _destroyDistance = 0.5f;
    private Vector3 _moveDirection;

    public float MoveSpeed { get => _moveSpeed; }
    public int Damage { get => _damage; }
    public float DestroyDistance { get => _destroyDistance; }
    public Vector3 MoveDirection { get => _moveDirection; }

    public ArrowModel (Vector3 moveDirection) {
        _moveDirection = moveDirection;
    }
}
```

Клас ArrowModel визначає основні властивості стріл, такі як швидкість руху (_moveSpeed), шкода (_damage), дистанція знищення (_destroyDistance) та напрямок руху (_moveDirection).

Рівень представлення відповідає за відображення ворогів (лістинг 3.3) і стріл (лістинг 3.4) гравцеві. У Unity це реалізовано через компоненти та об’єкти сцени, такі як спрайти та анімації.

Лістинг 3.3 – Клас EnemyView

```
public class EnemyView : MonoBehaviour {
    private Animator _animator;
    private EnemyModel _model;

    private const string IS_DEATH = "Death";
    private const string IS_RUN = "IsRun";
    private const string IS_ATTACK = "IsAttack";

    public void Construct(EnemyModel model) {
        _animator = GetComponent<Animator>();
        _model = model;
        _model.OnDied += OnDied;
    }

    private void Update() {
        _animator.SetBool(IS_RUN, _model.IsRun);
        _animator.SetBool(IS_ATTACK, _model.IsAttack);
    }

    private void OnDestroy() {
        _model.OnDied -= OnDied;
    }

    private void OnDied() {
        _animator.SetTrigger(IS_DEATH);
    }
}
```

Клас EnemyView прив’язаний до об’єкта ворога на сцені та оновлює його відображення відповідно до змін у моделі. Він використовує Animator для керування анімаціями ворога на основі його станів: біг (IS_RUN), атака (IS_ATTACK) та смерть (IS_DEATH).

Контролер обробляє взаємодію між моделлю та поданням, забезпечуючи логіку руху ворогів та стріл, їхню взаємодію та керування станами.

Лістинг 3.4 – Клас EnemyController

```
[RequireComponent(typeof(EnemyView))]
public class EnemyController : MonoBehaviour {
    private EnemyModel _model;
    private EnemyView _view;
    private MainBuilding _mainBuilding;

    public EnemyModel Model => _model;

    public void Construct(MainBuilding mainBuilding) {
        _model = new EnemyModel();
        _view = GetComponent<EnemyView>();
        _view.Construct(_model);
        _mainBuilding = mainBuilding;
    }

    public void StopAttack() {
        _mainBuilding.TakeDamage(_model.Damage);
        _model.IsAttack = false;
    }

    public void Death() {
        EnemySpawner.EnemyList.Remove(this);
        Destroy(gameObject);
    }

    private void Update() {
        float moveDistance = _model.MoveSpeed * Time.deltaTime;
        Vector3 moveDirection = FindTarget(_mainBuilding.transform);
        bool canMove = moveDirection.x < -0.01;
        _model.IsRun = canMove;

        if (canMove) transform.position += moveDirection * moveDistance;
        else if (!_model.IsAttack) Attack();
    }

    private Vector2 FindTarget(Transform target) {
        Vector3 movementVector = new Vector3(target.position.x - transform.position.x, 0,
0).normalized;
        return movementVector;
    }

    private void Attack() {
        _model.IsAttack = true;
    }
}
```

Клас EnemyController створює екземпляр ворога, задає його початкові значення та оновлює його стан під час гри. Контролер відповідає за переміщення ворога по карті, взаємодію з гравцем та іншими ігровими об’єктами. Він

використовує метод Construct для ініціалізації ворога та встановлення зв’язків між моделлю та поданням. Метод Update відповідає за оновлення стану ворога, включаючи рух до цілі та атаку.

Клас ArrowController (лістинг 3.5) створює екземпляр стріли та керує її польотом, перевіряючи на зіткнення з ворогами та завдання шкоди. Метод Construct ініціалізує стрілу, задаючи їй напрямок руху. Метод Update оновлює позицію стріли на кожному кадрі, обчислює її кут та перевіряє на зіткнення з ворогами. Якщо стріла потрапляє в ціль або виходить за межі екрану, вона знищується.

Лістинг 3.5 – Клас ArrowController

```
public class ArrowController : MonoBehaviour {
   private ArrowModel _model;

   private bool _isHit = false;

   private float _borderX = 8f;
   private float _borderY = 3f;

   public void Construct(Vector3 targetPosition) {
      Vector3 moveDirection = targetPosition - transform.position;
      _model = new ArrowModel(moveDirection.normalized);
   }

   private void Update() {
      if (_isHit) return;
      transform.position += _model.MoveDirection * _model.MoveSpeed * Time.deltaTime;

      float angle = GetAngle(_model.MoveDirection);
      transform.eulerAngles = new Vector3(0, 0, angle);

      foreach (EnemyController enemy in EnemySpawner.EnemyList) {
         if (Vector3.Distance(transform.position, enemy.transform.position) <
_model.DestroyDistance ) {
            enemy.Model.TakeDamage(_model.Damage);
            _isHit = true;

            ArrowDestroy();
            break;
         }
      }

      if (Math.Abs(transform.position.x) > _borderX || Math.Abs(transform.position.y) > _borderY)
Destroy(gameObject);
   }
```

```
    private float GetAngle(Vector3 direction) {
        direction = direction.normalized;
        float n = Mathf.Atan2(direction.y, direction.x) * Mathf.Rad2Deg;

        return n - 90;
    }

    private void ArrowDestroy() {
        Destroy(gameObject);
    }
}
```

Подання економіки реалізовано за допомогою Canvas та компонентів Text для відображення кількості золота та їжі. Спрайти використовуються для візуалізації будівель та веж на ігровому полі.

Створений демонстраційний прототип гри в жанрі Tower Defense на рушії Unity за допомогою C# реалізує основні елементи економічної моделі. Використання архітектури MVC для ворогів та стріл забезпечує чітку структуру коду та полегшує майбутнє розширення та підтримку проєкту. Прототип дає змогу гравцеві керувати ресурсами, будувати шахти, ферми та захисні вежі, що відповідає цілям гри – обороні поселення від ворогів.

3.3. Ігрова економіка в дії

Гра розроблена на основі економічної моделі, де гравцю потрібно захищати поселення від хвиль ворогів. Ігролад полягає в обороні поселення, яке атакують вороги хвилями на кількох етапах. Усього передбачено 5 етапів, кожен з яких складається з 8 хвиль ворогів. Вороги однакові й атакують головну будівлю в поселенні, б’ючи її в ближньому бою, коли добігають до неї.

Інтерфейс гри, зображений на рис. 3.7, включає верхню панель, яка показує стан здоров’я головної будівлі, кількість золота та їжі, а також їхній баланс, що відображає різницю між надходженням і витратами ресурсів. Знизу зліва розміщено показник прогресу хвиль, що дозволяє гравцю стежити за поточним етапом атаки.

Рисунок 3.7 – Вигляд розробленого прототипу гри

У центрі нижньої частини екрана розташовані кнопки для побудови споруд: золотих шахт, ферм та захисних веж. Поруч із ними праворуч знаходяться кнопки для купівлі апгрейдів для цих споруд. Кожна кнопка має позначку з ціною, необхідною для покупки.

На карті є порожні місця, що вказують на можливість побудови споруд. Коли гравець купує будівлі, вони з’являються на карті в цих порожніх місцях. Поселення та споруди розташовані ліворуч на екрані, тоді як вороги з’являються з правої частини й рухаються до поселення.

Гра побудована на механіці управління ресурсами, де гравець повинен ретельно балансувати між видобутком золота та їжі, побудовою нових споруд і захистом поселення від ворогів. Успішна стратегія залежить від ефективного використання ресурсів для зміцнення оборони і підтримки достатнього запасу ресурсів для подальшого розвитку.

ВИСНОВКИ

За результатами виконання кваліфікаційної роботи було розроблено ігрову економіку для демонстраційного прототипу гри, що базується на моделі, створеній за допомогою вебінструменту Machinations. Ігрова економіка включає базові ресурси, такі як золото та їжа, і їх взаємодію через побудову та утримання золотих шахт, ферм і захисних веж. Застосування ігрового рушія Unity в поєднанні з Machinations спростило процес розробки і дало змогу створити збалансовану економічну систему, що сприяє цікавішому ігровому процесу.

Також варто зазначити, що моделювання ігрової економіки в Machinations дозволило отримати глибоке розуміння взаємодії між різними елементами гри, що стало основою для подальшого вдосконалення ігрових механік. Зокрема, можливість адаптації моделі до змін та простота оновлень робить її гнучкою та перспективною для використання в інших ігрових проєктах.

Отже, кваліфікаційна робота демонструє важливість грамотного проєктування ігрових економік та їхньої інтеграції у процес розробки відеоігор. Отримані результати свідчать про перспективність подальших досліджень та удосконалення моделей ігрових економік для підвищення якості та привабливості ігор.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Kłuś P. The art of balance: understanding game economy design. crustlab. URL: https://crustlab.com/blog/understanding-game-economy-design/ (дата звернення: 23.04.2024).

2. Porokh A. What is video game economy design?. Kevuru Games. URL: https://kevurugames.com/blog/what-is-video-game-economy-design/ (дата звернення: 23.04.2024).

3. Paul McCarlie and Aaron Hunter. Using Game AI to Control a Simulated Economy. SciTePress - SCIENCE AND TECHNOLOGY PUBLICATIONS. URL: https://www.scitepress.org/Papers/2021/102123/102123.pdf (дата звернення: 18.05.2024).

4. 5 Basic Steps in Creating Balanced In-Game Economy | Room 8 Studio. URL: https://room8studio.com/news/5-basic-steps-in-creating-balanced-in-game-economy/ (дата звернення: 22.05.2024).

5. Game Currencies: Types and Usage. URL: https://jb-dev.net/2021/01/13/game-currencies-types-and-usage/ (дата звернення: 26.05.2024).

6. Michael Leibner. How we used Machinations to simulate the economy of IndusTree. Well Done Games. URL: https://well-done-games.com/blog/how-we-used-machinations-to-simulate-the-economy-of-industree (дата звернення: 26.05.2024).

7. The Designer's Notebook: Machinations, A New Way to Design Game Mechanics. Game Developer | Game Industry News, Deep Dives, and Developer Blogs. URL: https://www.gamedeveloper.com/design/the-designer-s-notebook-machinations-a-new-way-to-design-game-mechanics (дата звернення: 21.05.2024).

8. What is game economy design. URL: https://machinations.io/articles/what-is-game-economy-design (дата звернення: 10.05.2024).

9. What is Machinations? URL: https://machinations.io/docs/what-is-machinations (дата звернення: 26.05.2024).

10. Conor Stephens and Chris Exton. Measuring Inflation within Virtual Economies using Deep Reinforcement Learning. SciTePress – Science and

Technology Publications. URL: https://www.scitepress.org/PublishedPapers/2021/103928/103928.pdf (дата звернення: 26.05.2024).

11. The MMO Economist: AI Empowers Robust, Healthy, and Sustainable P2W MMO Economies. ACM Digital Library. URL: https://dl.acm.org/doi/pdf/10.1145/3589335.3648344 (дата звернення: 28.05.2024).